DOCUMENT RESUME

ED 326 390 SE 051 749

AUTHOR Reusser, Kurt; And Others
TITLE Tutoring Mathematical Word Problems Using Solution Trees: Text Comprehension, Situation Comprehension, and Mathematization in Solving Story Problems. Research Report No. 8.
INSTITUTION Bern Univ. (Switzerland).
PUB DATE Apr 90
NOTE 39p.; Paper presented at the Annual Meeting of the American Educational Research Association (Boston, MA, April 16-20, 1990).
PUB TYPE Speeches/Conference Papers (150)

EDRS PRICE MF01/PC02 Plus Postage.
DESCRIPTORS Abstract Reasoning; Artificial Intelligence; Cognitive Ability; *Computer Uses in Education; Critical Thinking; *Elementary School Mathematics; Elementary Secondary Education; Foreign Countries; Mathematical Logic; Mathematics Education; *Problem Solving; *Psychoeducational Methods; *Secondary School Mathematics; Thinking Skills; *Tutorial Programs; *Word Problems (Mathematics)

ABSTRACT

The main concern of this paper is on the psychological processes of how students understand and solve mathematical word problems, and on how this knowledge can be applied to computer-based tutoring. It is argued that only a better understanding of the psychological requirements for understanding and solving those problems will lead to instructional improvement. Introduced first are the major points of a psychological theory of strategic and goal-directed understanding and solving mathematical situation problems derived from work on discourse processing and on work on cognitive simulation related to word problems. Secondly, the psychological and didactic principles, and the techniques used for the development of a computerized tutor designed at helping students to solve complex, verbally stated, mathematical problems are discussed. Developing partly intelligent tutoring systems may not only serve as a means to provide children with the kind of interactive problem solving and learning environments and instructional tools which have been described as didactic models of scaffolding and coaching, reflective learning, or of cognitive apprenticeship, respectively, but also serve as a means to the more fundamental educational goal of becoming an autonomous or self-directed learner. The computer tutoring program, called HERON, uses an iconic form of problem representation called solution trees or conceptual planning trees, a graphical network format of semantic concepts and mathematical operations, intended to capture the hierarchical structure of solution plans of multi-step mathematical situation problems. A narrative and graphical description of the program is included. (KR)

ED 263390

Textverstehen, Situationsverstehen und Mathematisierung beim Lösen von Textaufgaben[1]

Universität Bern, Abteilung Pädagogische Psychologie

U.S. DEPARTMENT OF EDUCATION
Office of Educational Research and Improvement
EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)
✓ This document has been reproduced as received from the person or organization originating it.
☐ Minor changes have been made to improve reproduction quality.
• Points of view or opinions stated in this document do not necessarily represent official OERI position or policy.

"PERMISSION TO REPRODUCE THIS MATERIAL HAS BEEN GRANTED BY
Kurt Reusser
TO THE EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)."

Forschungsbericht/research report

No. 8

Tutoring Mathematical Word Problems Using Solution Trees

Kurt Reusser, Alexander Kämpfer, Markus Sprenger, Fritz Staub, Rita Stebler, and Ruedi Stüssi

University of Bern, Switzerland

April 1990

Text comprehension, situation comprehension and mathematization in solving story problems

University of Bern

SE 051 749.

[1] Unterstützt durch den Schweizerischen Nationalfonds zur Förderung der wissenschaftlichen Forschung: Projekt Nr. 1.448.-0.86.

Tutoring Mathematical Word Problems Using Solution Trees.[1]

Kurt Reusser, Alexander Kämpfer, Markus Sprenger,
Fritz Staub, Rita Stebler, and Ruedi Stüssi,

University of Bern, Switzerland.

The main concern of this paper is on the psychological processes of how students understand and solve mathematical word problems, and on how this knowledge can be fruitfully applied to computer-based tutoring. It is argued, that only a better understanding of the psychological requirements for understanding and solving those problems will lead to instructional improvement. We first introduce major points of a psychological theory of strategic and goal-directed understanding and solving mathematical situation problems derived from work on discourse processing (Van Dijk & Kintsch 1983) and on work on cognitive simulation related to word problems (Kintsch & Greeno 1985; Cummins, Kintsch, Reusser & Weimer 1988; Reusser 1985, 1989a,b). Then, we discuss the psychological and didactic principles, and the techniques used for the development of a computerized tutor designed at helping students (from elementary level to high school) to solve complex, verbally stated, mathematical problems. We will argue that developing partly intelligent tutoring systems not only may serve as a means to provide children with the kind of interactive problem solving and learning environments and instructional tools that have recently been described as didactic models of scaffolding and coaching (Palincsar & Brown 1984; Brown and Palincsar 1989), reflective learning (Collins & Brown 1988), or of cognitive apprenticeship, respectively (Collins, Brown & Newman 1989), but also as a means to the more fundamental educational goal of becoming an autonomous or self-directed learner (Beck 1989; Reusser 1990).

As it is known from educational practice and from numerous studies conducted over the past decade, mathematical word problems are difficult for students at all grade levels. They are widely used in most (western) school systems in order to test applied, or situated, mathematical thinking and problem solving in critical contexts

[1] Paper presented at the AERA meeting in Boston, April, 16-20, 1990.

of student's educational careers. Moreover, the solving of verbally stated mathematical problems is often seen as a suitable measure of how well, in general, students are able to understand, structure, or model complex situations, how they are able to do planning, abstraction, or reasoning - skills that are commonly labeled "metacognitive" today.

Professional teaching of any kind of subject matter needs to be based - among other requirements - on a sound psychological and didactical analysis of the curricular task at hand: both as a learning product and as a (knowledge-acquisition, problem solving or learning) process. Related to word problems: Anyone, who wants to teach children how to understand and solve mathematical word problems, needs to know by which processes or strategies, and mediated by which mental representations, mathematical information is abstracted from verbal problem statements.

From Text to Situation to Equation: A Theory of Understanding and Solving Mathematical Word Problems

Before describing the development of our tutor, we briefly outline and discuss some previous work on word arithmetic problems from which essential parts of our tutoring approach are derived. Extending the work of Aebli (1980), Reusser (1984) and Kintsch & Greeno (1985), Reusser (1985, 1989b) developed a cognitive simulation model of understanding and solving elementary word arithmetic problems based on a decomposition analysis of the involved comprehension skills. The rule-based model takes as input elementary addition and subtraction word problems, "understands" and solves them reliably by means of various types of strategies. According to the underlying theory, word problem solving consists of a strategic process (cf. van Dijk & Kintsch 1983) of translating a fuzzy natural-language problem statement into the canonical structure of a formal equation. This includes the construction of at least four interrelated and mutually constraining mental representations or levels of comprehension: the *textbase* as a propositional representation of the textual input, the *(episodic) situation model* as a representation of the situation the text is about, the *(mathematical) or formal problem model* as the abstract gist of the problem situation, and *a numerical equation* (Figure 1). In order to

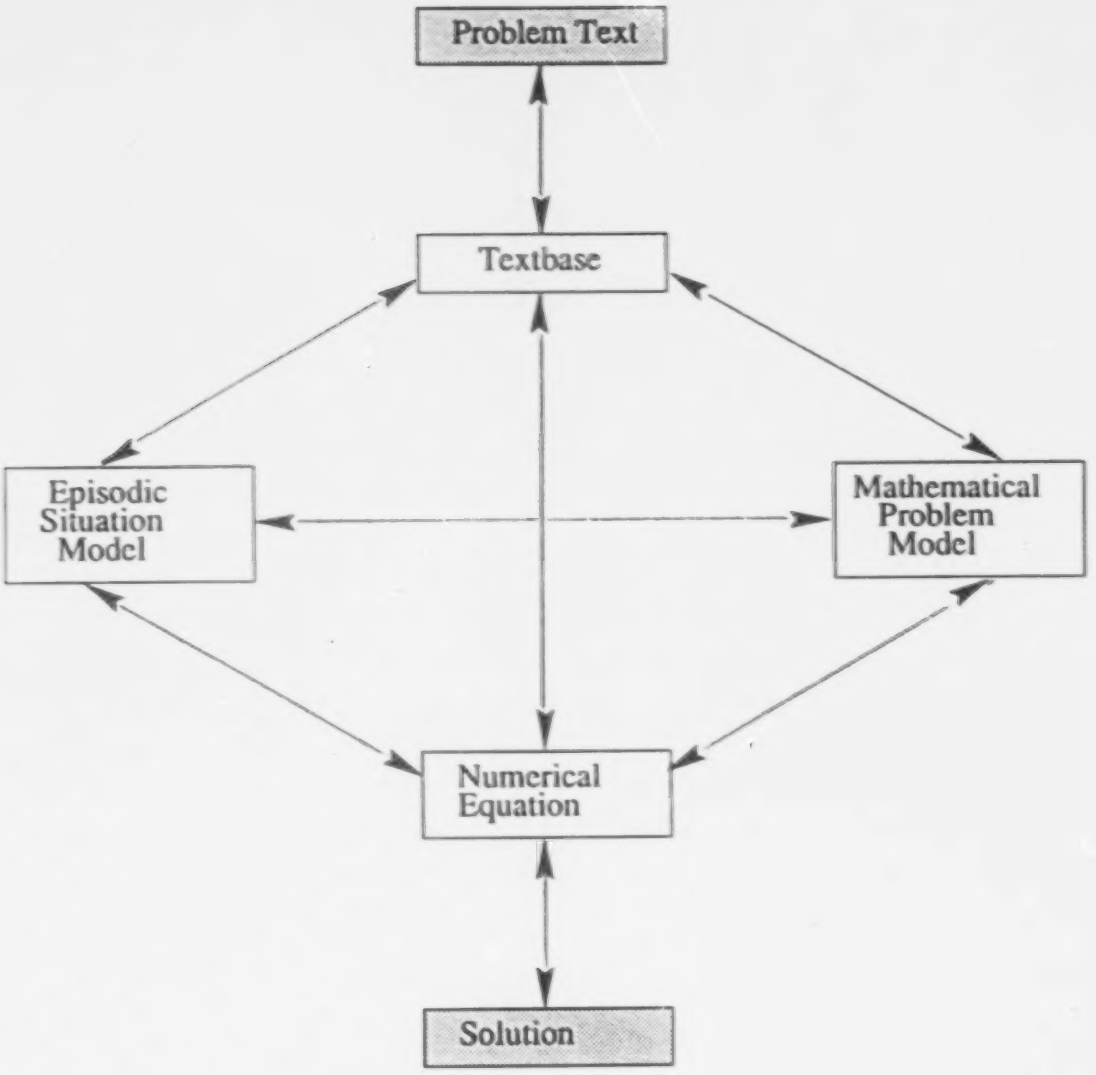


Figure 1. Levels of mental representation in understanding and solving mathematical word problems (REUSSER 1985, 1989).

achieve the right *mapping between text and equation, between situation and formal problem model*, the ideal didactic or teaching sequence with respect to the solving of particular problems is to go through these levels of comprehension. Empirical subjects, however, often deviate in characteristic ways from this pattern. On one hand, experts or skilled students in a domain frequently skip the construction of a sound informal (episodic, real-world) situation model. Their powerful and context appropriate schematic and calculational strategies allow them to get directly at the abstract problem model - or even at the equation or numeric solution. Weaker or novice students, on the other hand, who often adopt similar strategies of "direct translation" (see Bobrow 1964) by just trying to to jump from text to equation (plugging numbers into equations), or by doing some magic number work, will fail with the same task of mathematization. Their lack of powerful domain knowledge does not allow them to immediately "see" the right mathematical schema, or to trigger any smart super-strategies (as e.g. the transfer-in/out super-schemata in the model of Kintsch & Greeno 1985) relating the situation described in the problem text to a mathematical problem model or even an equation. They need to carefully analyse, step by step, the problem in everyday concepts, to work bottom up, i.e. to establish a concrete, intermediary (non-mathematical) situation model before they can build a (mathematical) problem model that represents nothing more but the goal-relevant, quantitative and abstract gist of the problem episode.

As predicted by the theory, and as found in a series of empirical studies, understanding mathematical story problems is a complex and knowledge intensive inferential and strategic process which requires skillfull interaction of all kinds of knowledge including linguistic, situational and mathematical knowledge. The basic psychological and instructional hypotheses associated with Reusser's model says that, for most problem solvers, linguistically cued situational and mathematical understanding is not - and in an instructional context: *should not be* - a superfluous but a necessary and helpful achievement. From the fact that story problems are, in general, much more difficult than comparable numerical problems, it can be concluded that factors other than mathematical skill provide a major source of difficulty to word problems (cf., eg., Hudson 1983; De Corte, Verschaffel & De Win 1985). Their mathematical structure is merely one constraining factor of getting at the right arithmetic strategies. As Cummins et al. (1988) and Reusser (1989a) argue, com-

prehension failures with regard to various textual and situational factors which constitute the overall presentational structure or setting of word problems, greatly affect their difficulty and the ways in which students solve them.

Didactical Principles of (Un)Intelligent Tutoring

There is no maschine-based tutoring without a theory of teaching. Work on tutoring systems should be based both on research in *didactic or instructional theory* and on *research in cognitive psychology*. Related to tutoring of mathematical word problems: Who designs a computer tutor not only needs to know the psychological processes and strategies employed by students of different ability levels in understanding and solving word problems, but also how to teach these processes effectively. Thus, good computer-tutors need to exhibit instructional qualities or didactic skills, at least partly similar to human expert teachers related to a particular domain.

However, with Scardamalia & Bereiter (1987), Resnick & Johnson (1989), and Nathan, Kintsch & Young (1990), we are sceptical about truly *intelligent* or *adaptive* tutoring systems based on full system-control and deep student modeling. In a narrow and demanding sense intelligent tutoring has come to mean the computer to function as an intelligent substitute for a human teacher, as an (even better) adaptive system being not only able to perform on-line modeling of the student's knowledge and thinking but also as to perform cognitive diagnosis, i.e., to infer - on the basis of a constantly tuned student model - a persons cognitive states from his performance (cf. Ohlsson 1986). But given the current state of the art, mental diagnosis-driven tutoring based on *cognitive simulation*[2] is not possible in open-ended domains, as e.g., word problems, where language- and world knowledge-based reasoning, or natural language capabilities together with enormous amounts of world knowledge, are required. Furthermore, one can ask, as Scardamalia & Bereiter (1987) do, how reasonable such a system would be: "For it is not the computer that should be doing the diagnosing, the goal-setting and the planning, it is the student. The computer environment should not be providing the knowledge and intelligence to guide learning, it should be providing the facilitating structures and tools that

[2] in the sense of the cognitive simulation work on elementary word arithmetic problems mentioned above.

enable students to make maximum use of their own intelligence and knowledge" (5).

Intelligent tutoring systems, where a machine tailors its instruction to a student on the basis of a fine-grained, individually constructed mental model, may be seen as a long-term goal. But even if it was judged as realistic in principle (cf. Winograd & Flores 1986), and as feasible to be achieved in the future, it would be only one of the ways of applying advanced computing technologies to education.[3] From a didactic point of view, there are alternatives to support and facilitate human learning and problem solving through interaction with a computer. The most sensible one might to be viewing machine-supported tutoring under the ultimate goal of becoming a virtually autonomous and reflective learner and thinker. This does not require truly "intelligent" but flexible, intelligence-supporting didactical tools, where intelligence is seen as being located primarily in the learner and not in the computer. One should conceive of computers, therefore, not primarily as substitutes for intelligent students and teachers, but as intelligence-extending tools (Resnick & Johnson 1989), as didactical instruments in learning and teaching environments which are directed - to the greatest possible extent - at the fostering of learner autonomy and self-regulation.

Such an alternative view of computers as *cognitive tools for intelligent learners and virtually autonomous problem solvers* can be situated within the epistemological and didactical framework of models and metaphors currently being discussed in applied metacognitive research, as e.g., in the Vygotskian inspired models of coaching and scaffolding (Palincsar & Brown 1984; Aebli, Ruthemann & Staub 1986; Brown & Palincsar 1989), of cognitive apprenticeship (Collins, Brown & Newman 1989), of procedural facilitation (Bereiter & Scardamalia 1987), of reflective learning (Collins & Brown 1988), or of autonomous and self-directed learning and problem solving (Bruner 1986; Beck 1989; Reusser 1990). According to these ideas, learners do not develop their cognitive and professional abilities as solo-learners (Bruner 1986), i.e. purely on their own[4], but by getting support - procedural

---

[3] see Resnick & Johnson (1989) for a broader picture of interactive computer applications in education.

[4] cf. the concepts of "learning by inventing" (Piaget 1973), or "learning by discovery" (Bruner 1961).

facilitation - from parents, teachers, master-craftsmen and further significant others (Mead 1934). Computers as educational devices for intelligent learners may contribute to this overall process of instructional scaffolding as functional parts of didactical environments (where the teacher still plays a fundamental role), and within these environments, as supplying students with convenient computer-assistance for more effective problem solving and thinking.

The tutoring approach that we have adopted in our work, can be characterized by a number of design principles[5] providing the conceptual basis of the system development to be outlined below. The principles are not technical specifications of particular systems but rather cognitively based didactical propositions describing the proposed capabilities of such systems.

1. *Computer-supported tutoring should provide students with effective scaffolding for understanding, planning and conceptual representation in problem solving.* Comprehension activities, goal-setting, planning, inferencing, reflecting, and self-monitoring - processes that students need to exercise in order to learn (meta)cognitive skills - depend on the ability of drawing on one's own resources for attacking problem situations. Computer systems which are well-equipped to do (and undo) things as generating icons, as presenting, touching, linking, placing, dragging, storing, selecting, and retrieving information, or at bookkeeping, and monitoring the action sequences of its users, can provide tools to facilitate and to guide students in these activities (procedural facilitation). Tutoring is considered successful to the degree it maximizes student's sense of control and to the degree student solutions are self-generated (Anderson 1989).

2. *Good tutors should provide students with intelligible, domain-appropriate formats of problem representation which are based on cognitive task analysis*

[5] the value of our principles might not be restricted to the design of our tutor. They contrast in certain ways to principles outlined earlier by Anderson, Boyle, Farrell & Reiser (1984), or by Ohlsson 1986, addressing the question of cognitive diagnosis. In part, our principles are similar to those laid out by Scardamalia & Bereiter (1987), and to principles underlying the ANIMATE tutor of Nathan & Kintsch (1990).

and which constrain the construction of formal problem models. To understand a text problem in order to solve it, means to form a mental model, i.e., to find a form of representation intermediate to the textbase and the underlying abstract structure (e.g. an equation). A wisdom of classical theory of problem solving says that a properly represented problem is half a solved one. Because good representations help to organize a problem structure around the salient properties of its deep structure, they can greatly affect problem solving efficiency (Clancey 1988)[6]. Thus, to offer a theoretically intelligible representation format for a class of problems provides students with the task-appropriate perspective for building adequate (formal) problem models during the problem comprehension process. Providing deep structures of planning and problem representation is more than let the students "learning by doing" (Sleeman & Brown 1982), a tutoring approach that has been criticized repeatedly (cf. Derry 1989). Graphic or symbolic representations (networks, tree structures, algebraic schemata, animation) based on psychological and instructional task analysis, constrain the construction of situation models. Given a form(at) of a solution, the many possible ways of representing a problem as a knowledge structure and of solution planning, are reduced to a range of a few well-designed alternatives.

3. *Tutoring systems should help students to become more reflective by making implicit processes explicit, and by reducing the load of working memory.* Covered or hidden knowledge construction activities of students should be turned into explicit or overt processes - identifiable both for the students and for the teacher. This can be done by allowing, or enforcing, students to do the construction work using specific, task-appropriate symbolic or graphical formats or schemata (cf. principle 2). What is mentally constructed in the head of the problem solver, should - to as great an extent as possible - be forced to be

[6] For this reason, good representational schemata which facilitate the capturing of the essential structure of a problem, and which are used by experts, should be taught to students (cf. Larkin et al. 1980)

turned overt, as comprehension or solution path, so it can be recorded and analysed (and even be re-played) as a sequence of temporal events. Log file data with further process characteristics may complement the construction process data. Moreover, the difficulty of holding a great deal of information in working memory often causes errors. Therefore, good tutors are to be good aids to memory. This is possible by externalizing much of working memory on the screen.

4. *Flexible tutors should students allow to get graduated help but only as much as they really need.* There is a delicate balance in teaching between too much and too little help to students. Good teachers follow the didactic "principle of minimal help" (Aebli 1961). Since in the near future, computer tutors are hardly to become intelligent with respect to demanding student modelling (a prerequisite for individually customized guidance), they should nevertheless provide various forms and degrees of help. But they should leave it to the student's cooperation in learning, to use or seek only as much help from the system as he/she needs. Students should decide on their own on the grain size of the explanatory help, or on the grain size of instructional hints they want to get.

5. *A tutoring system should promote the use of comprehension-related, general and domain-specific strategies. Together with the representational formats used in a system, theses strategies are the cognitive tools of thinking and problem solving.* If problem solving means to acquire, identify and select representation and planning strategies tuned to a specific domain or class of problems, students must be provided with and encouraged to use them. Examples of strategies for the word problem domain are: Analyse the problem question and determine the form of the unknown; separate mathematically relevant from irrelevant information; make a solution plan using a given graphical format.

6. *A tutor does not necessarily need to know in a deep way what the student is thinking*[7]*, but what the structure of the problem is, and what the student is doing while solving it.* Since, for domains which require large amounts of world knowledge, cognitive diagnosis is beyond the state of the art, detailed student modeling is hardly feasible.[8] However, in order to interact with the student, i.e., to provide him with some customized feedback, motivational encouragements, hints and help, a tutor should be able to understand the overt behavior of a student on the basis of a solid decomposition analysis of the task. Thus, tutoring design should be based, in the first place, on the conceptual analysis of the skill to be taught. The tutor should know mature (expert) and less mature models of the processes and representations to be taught or conveyed to students, so he can map their overt construction activities into them.

7. *A tutor should provide feedback on errors*. Making errors is an inevitable part of learning. Since learners can become highly confused and demoralized by undetected errors (Anderson et al. 1984), feedback must be provided - either immediate or delayed.[9] A convenient way of providing tutorial feedback is, as Nathan, Kintsch & Young (1990) do in their tutoring of distance-rate problems, let the student run an animation (time varying computer graphics) on the basis of his problem model so that he may decide on his own upon its correctness. In other classes of problems where animation does not work, feedback on errors must be based on an explicit conceptual analysis of the task or solution space which allows to observe when and how the knowledge-construction activity of a particular student deviates from a predetermined set of solution paths (cf. Derry & Hawkes 1989).

---

[7] i.e., what his/her cognitive states are, or "to represent the student as a production set" (Anderson et al. 1984, 4).

[8] see for a related argumentation Nathan & Kintsch & Lewis (1988) and Derry & Hawkes (1989. Derry & Hawkes write with regard to error modeling: "Deep modeling of procedural bugs is computationally intractable for complex problem domains, and we do not believe it is required for effective cognitive apprenticeship" (33).

[9] see Nathan & Kintsch (1990) for a discussion of either type of feedback in computer-based tutoring.

8. *Computer-based cognitive tools should not be seen as isolated teaching tools but rather as didactical elements of a broader classroom learning and teaching environment.* There is little hope that near future generations of effective computer learning tools will function as teacher-independent knowledge acquisition or problem solving systems, but rather as didactical teaching aids or tools which have to be embedded in a rich network of didactically reflected cognitive activities and interactions of its intelligent users.

HERON: Tutoring Mathematical Word Problems using solution trees as a format for problem representation and strategic planning

HERON - as we call our system[10] - is based on the theory of word problem comprehension and mathematization proposed in Reusser (1985, 1989) extending the work of Kintsch & Greeno (1985). According to this theory, the solving of word problems is viewed as following four mutually constraining steps: text comprehension (forming a textbase), qualitative, or situation-directed comprehension or reasoning, mathematical problem model formation, and generation of an equation. However, for reasons outlined in the above sections, HERON is not a cognitive simulation model in the strict sense but rather a *computerized didactic tool for facilitating and fostering self-directed and comprehension-based learning and problem solving*. The system is designed to aid students of grade levels 3 thru 9 in understanding and solving complex mathematical situation problems[11] by providing support in text comprehension and in building mathematical problem models.

This is done by means of two kinds of *didactical strategies* and with the help of *solution trees* or *conceptual planning trees*, a proposed basic format of conceptual representation which is intermediate to the text and the underlying equation. The first kind of strategies treats the problem text essentially "as a text" denoting an episodic action structure. Text comprehension strategies guide the analysis of the problem text in the direction of the building of a non-mathematical situation model. The second kind of strategies which are directed more towards the mathematical

---

[10] after the greek mathematician HERON of Alexandria (appr. 100 BC) who created among the first mathematical word problems still being found in modern mathematical text books.
[11] We are following a similar design philosophy as Nathan & Kintsch (1988;1990).

understanding of the problem, supports the construction of a mathematical problem model and the derivation of an equation from it.

The system gives the student a fair amount of interactive flexibility and a sense of control in representing the problem, and in (self-directed) planning the mathematical solution.

Solution trees

First, we shall present a discussion of the principal representational means our tutorial system is based on. Even if language provides us with the single most important means or medium (Bruner 1966) for communication and the representation of meaning, certain kinds of information can be represented or expressed better by pictorial or iconic means. To build, e.g., a situation model while solving a mathematical word problem, has something to do with the translation of the propositional information stated in the problem text into an adequate, imagery-like (pictorial), actional,[12] or schematic, i.e. more abstract, ultimately numeric representation. As educators know since centuries (Comenius 1659), good representational formats are not only of high educational value but also should be conveyed, as targets of teaching, to young people.

HERON is using an iconic form of problem representation called *solution trees* (Aebli, Ruthemann & Staub 1986; Derry & Hawkes 1989)[13], or *conceptual planning trees*, a graphical network format of semantic concepts and mathematical operations, intended to capture the hierarchical structures of solution plans of a very broad range of complex (i.e. multi-step) mathematical situation problems.

---

[12] allowing a student to mentally or effectively act out a problem situation, Bruner (1966) hat drawn our attention to the fact that some knowledge and some communication can be realized by means of action (Aebli 1981).

[13] Derry & Hawkes (1989) developed a very similar approach of tutoring mathematical word problems. They also use solution trees as a basic problem representation, and they follow a similar tutoring philosophy. We have developed our theoretical ideas, as well as our tutoring approach, completely independent of their work. Two graduate students from Bern were very surprised when they became aware of the similarity of our approaches at the Fourth International Conference on AI and Education in Amsterdam, 1989.

Moreover, solution trees allow to trace the high-level structures and solution paths that are built by students while solving a problem.

The concept of solution trees[14] was derived from the earlier notion of "Simplex-Complex structures" (Breidenbach 1963; Fricke 1987) developed in mathematics education of the sixties. The representational format of solution trees is used in Swiss and German math books. They were found to function as an intelligible and effective (Aebli et al. 1986) format of mathematical representation.

Solution trees are hierarchical network structures of mathematical concepts and relations. They consist, as their building blocks, or conceptual molecules, of *domain-specific relational schemata*[15] or *specific domain-relations*, each schema being a subgoal in an arbitrary complex hierarchic solution plan and consisting of a content-specific, *mathematical relation* of two qualitatively and numerically specified *situation elements or units* (Reusser 1987; 1990) allowing to compute a third, unknown, unit. Each unit itself consists of three subfields of information: a field for the *numerical value*, which can be unknown (as goal or as an intermediary subgoal of comprehension and computation), a field for the *unit of measurement*, and - most important - a field for a *textual label defining the situational role* of an unit, or linking the quantitative information to a qualitative situation model.

The graphical form of three situation elements forming a relational schema is depicted in Figure 2. Table 1 shows a sample set of specific domain relations (Sachverhältnisse; Selz 1922) frequently occurring in word problems. The conceptual complexity of these relations is a major source of the difficulty of word problems requiring worldknowledge-based, situation-directed reasoning.

Solution trees are formed through a forward- or backward-inferencing, constructive activity. The goal or subgoal-driven activity proceeds, until the solution anticipated by a given or self-generated question, can be computed by a final schema.

The concept of solution trees is more than a derivative of teaching experience and intuition. It is in many ways relatable to psychological theory on discourse and

---

[14] they are also called "calculation trees" (Rechenbäume)
[15] Selz (1922) called them *Sachverhältnisse*, relations between subject-matter elements.

Figure 2. Three specific domain-relations mirroring three nascent versions of a relational schema: (a) and (b) are incomplete schemata, (c) is completed.

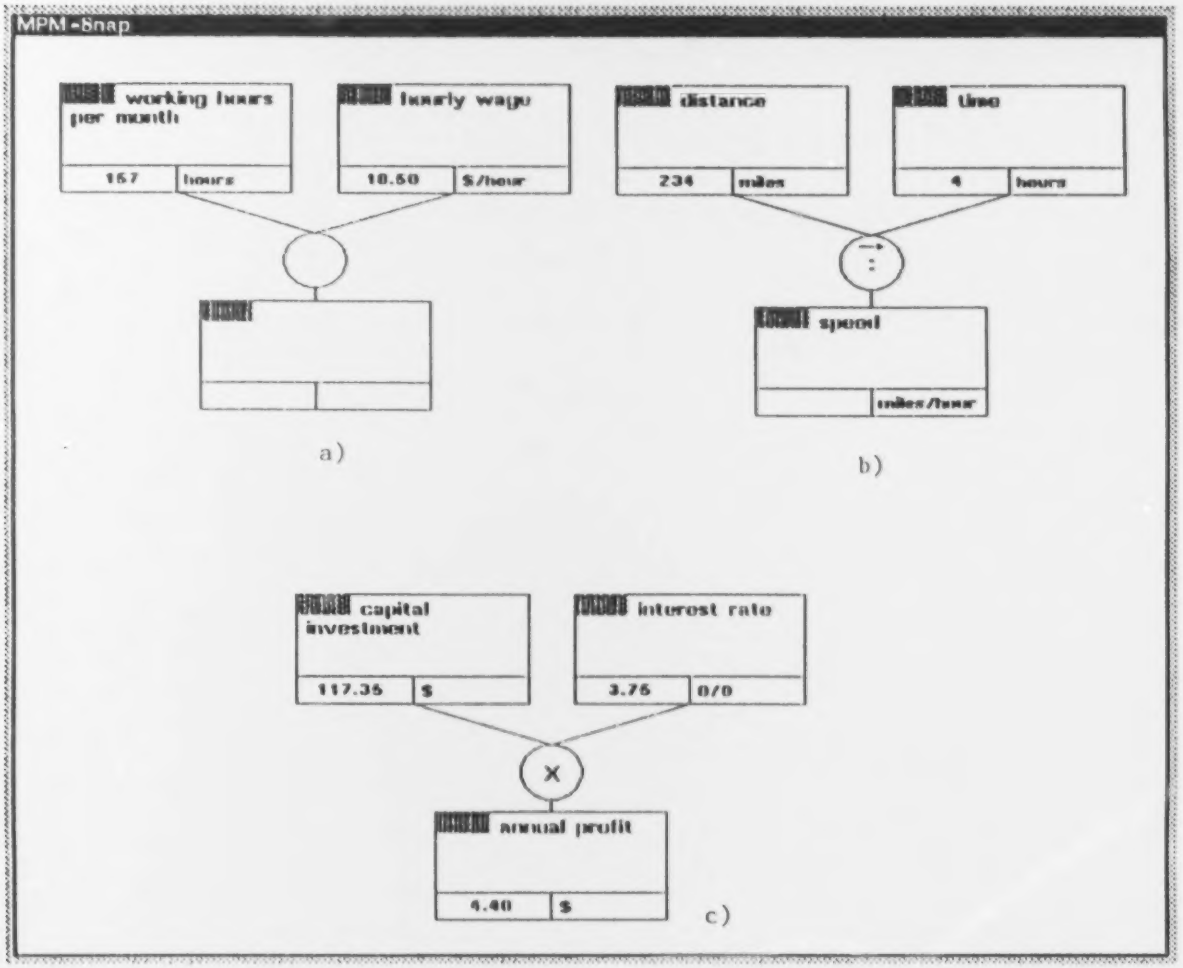


TABLE 1.

Sample domain-specific relations forming the basic conceptual schemata of HERON. A schema is always relating (R) three domain-specific concepts in a mathematically meaningful way.

R (amount 1, amount 2, total amount)

R (unit-price, amount, total cost)

R (distance, speed, time)

R (amount per time, rate, time)

R (list price, handling/postage, fixed cost)

R (sale price, discount, initial price)

R (capital investment, interest rate, annual profit)

R (volume, mass, density)

R (salary, time, hourly wage)

R (saving goal, rate of saving, time to wait)

mathematical comprehension, and on problem solving. According to an epistemological assumption of Piaget (1950), development of mathematical thinking is deeply embeded into the development of acting, of qualitative reasoning and reflection about everyday situations. Mathematical thinking emerges from acting. Mathematical operations can be psychologically seen as developmental derivatives of sensorimotor actions. Aebli (1980), who has further elaborated Piaget's concept of operation, defines them as "abstract actions": Both to act and to operate means (if the operation is performed consciously, or comprehension-based) to establish qualitative, and in the case of mathematics, quantitative (or set) relations between elements of a qualitative, or episodic situation structure. Textually stated mathematical word problems are situation structures carrying a *latent mathematical structure*. To understand and solve a text problem means to focus on its quantitative structure, driven by a problem question. But this is exactly what solution trees force students to do: to focus on the mathematically relevant properties of a situation structure. Because solution trees provide the *schematic form* for the actional, or - in the sense of Piaget - operative, mathematical deep structure of a wide class of problem situation, they are a valuable representational means for its abstractive construction. Solution trees mediate language comprehension and world knowledge-related, qualitative reasoning to quantitative reasoning, or to mathematical thinking. They *schematically anticipate* (Selz 1922) essential aspects of mathematical problem models, or they provide a perspective for their construction. While the mathematical understanding of the problem situation takes shape in the head of the problem solver, the schematic form of the solution tree serves as a constraint that must be satisfied by its overt construction - on paper, or on the screen of a computer.

HERON: The prototype system

HERON is a computer system that supports students in understanding and solving complex word problems. The system is particularily designed to aid students with three types of activities which are represented by kinds of didactical strategies: the *comprehension of a problem text*, the *construction of a solution tree* based on the comprehension of the situation denoted by the problem text, and the *derivation of a corresponding numerical equation*. HERON is able to facilitate the solving of any

problem that can be represented by solution trees. There is no principle, only a practical limitation of task complexity: the size of the computer screen. There are two implementation versions of HERON under development: One is written in Loops and runs on a Xerox 1186 workstation (further referred to as the XEROX-version); another version designed for the experimental use in classrooms is written in C and runs on IBM-type computers (further referred to as the AT-version).

Only little instruction is needed to achieve almost full use of functions with HERON. It takes third graders about 20 minutes to get familiar with the entirely mouse-driven interface of the system. Most commands and graphical tools are available as buttons displayed on the screen. Some important menus, as, e.g. for the filling in text labels into solution trees, can be activated by pushing a mouse button in an appropriate, active region of the screen.

By activating the screen-displayed buttons or the menus with the help of a mouse-button, students can

- select or quit tasks,
- select, link and instantiate building blocks of solution trees: empty situation units, line segments, mathematical operators, numbers, units of measurements, or whole relational schemata consisting of three situation units,
- attach textual labels to situation units (by selecting or composing them from a menu),
- correct errors on any element, label, link, number and operator,
- display the problem text on the screen or hide it,
- move back and forth beween problem text and solution tree,
- flexibly move and re-place graphical units on the screen,

- get support for the (partial) derivation of an equation matching the constructed solution tree,

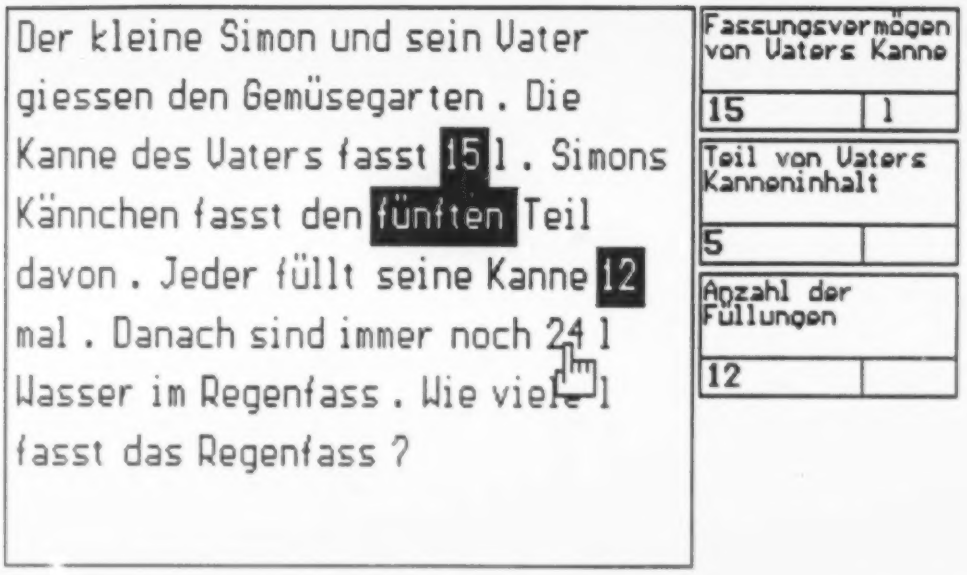
- let the system calculate any unknown quantity of the tree.

In order to explore the functionality of the system, we follow an example of how students can use it. Previous to the construction of anything, the student selects a problem and reads it.

> *Little Simon and his father are watering their vegetable garden. The father has a 15-liter watering can. Simon's can hold one fifth of that. Both fill their cans 12 times. After that, there are still 24 liters in the waterbutt. How much water can the waterbutt hold?*

After a reading phase, the student is asked to identify relevant quantitative information in the problem text (Figure 3). This is done by highlighting numbers or number-placeholders in the text with the mouse-cursor. When a piece of numerical information is selected, the system automatically creates a graphical situation unit where the selected number is already filled-in. The student is asked to fill in a unit of measurement (e.g. liters) and a textual label (e.g. content of father's can). The label can be selected as a whole from a menu, or it can be composed from a list of wording elements from a menu. After the situation unit has been completed, a new piece of information is selected in the text. According to how the system is initialized (e.g. for very weak or novice students), the student gets an error message if he or she selects a piece of irrelevant information from the text, or if a relevant piece is forgotten in the selection. In the standard running condition, the student decides on his/her own when to stop selecting numbers and creating situation units.

After getting some situation elements ready, the student can start planning and constructing a tree-structure of the solution (Figure 4). In order to create and instantiate a first relational schema, e.g., in order to achieve the subgoal of computing the content of Simon's can, the units labeled "content of father's can" and "part of father's can" are selected with the mouse, moved to the upper left corner of the screen and linked together. The latter is done by selecting, placing and linking an


Der kleine Simon und sein Vater giessen den Gemüsegarten . Die Kanne des Vaters fasst 15 l . Simons Kännchen fasst den fünften Teil davon . Jeder füllt seine Kanne 12 mal . Danach sind immer noch 24 l Wasser im Regenfass . Wie viel l fasst das Regenfass ?
Fassungsvermögen von Vaters Kanne
15
l
Teil von Vaters Kanneninhalt
5
Anzahl der Füllungen
12

Little Simon and his father water their vegetable garden. The father has a 15-liter can. Simon's can holds one fifth of that. Both fill their cans 12 times. After that there are still 24 liters in the waterbutt. How much water can the waterbutt hold?

Figure 3. Identifying relevant problem information

empty operator node (circle) and two line segments. With the selection of an operator node, a resulting empty box is automatically generated by the system.[16] Before the user can fill in the unit of measurement and the label into the emerging third element, he/she is asked to select the appropriate mathematical operation (":") from a menu which appears when pushing the mouse on the operator node. Now the student can fill in the label ("content of Simon's can") and the unit of measurement ("liters") and decide if he/she wants to calculate the subgoal immediately. The calculation can be done with or without system support: e.g., clicking on the numerical field lets the system calculate and display the correct result in the numerical field.

After each relational *schema* being instantiated, its resulting element can be used to generate new schemata and to achieve further subgoals. It is up to the student which comprehension paths to follow, that is, how to navigate thru the problem space. In order to determine the intermediate level subgoal of the "total amount of carried water by both father and son", there are two main paths open to follow. In the first path (depicted in Figure 4 or 5), an additive schema is first generated which computes the "content of both watering cans", and then a muliplicative schema is instantiated computing the "total amount of water". In the second, slightly more complicated path (depicted in Figure 6), the order of mathematical operations is reversed: The instantiation of two multiplicative schemata

MULTIPLY (number of fillings, content of father's can)
MULTIPLY (number of fillings, content of Simon's can)

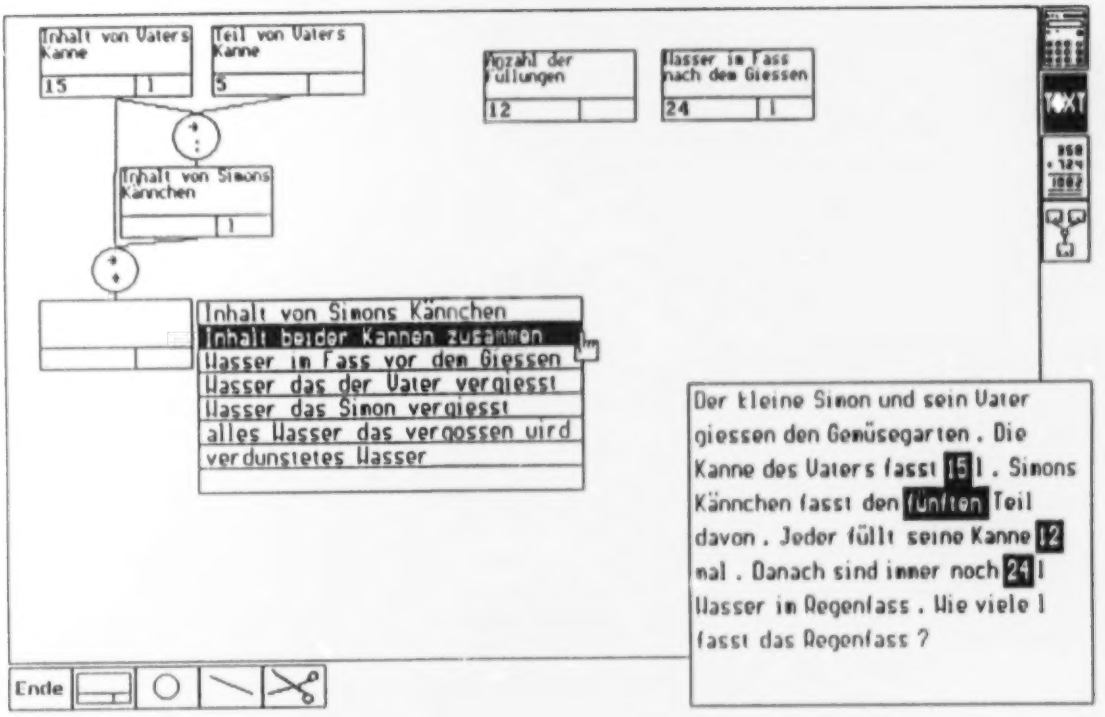
is followed by an additive schema

ADD (amount of water carried by the father, amount of
water carried by the son)

leading to the same intermediary result (216 liters). Relating the "total amount of carried water" to the "amount of remaining water in the waterbutt" (24 liters) and computing the final schema leads to the final result of the "amount of water in the full waterbutt" (240 liters).

---

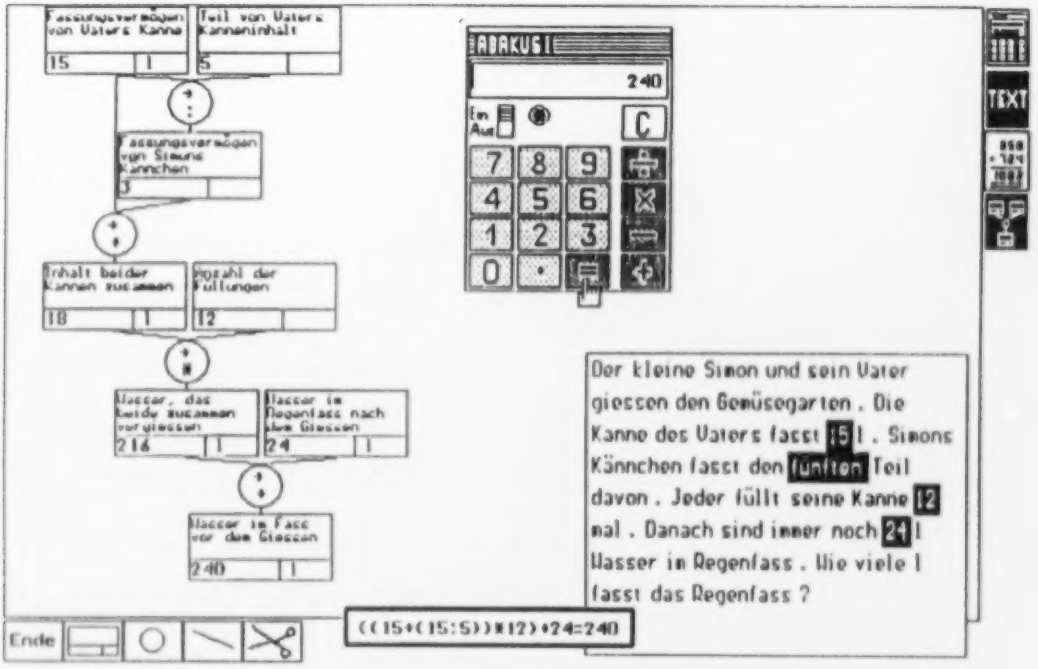
[16] To every pair of situation elements linked by an empty operator slot (circle), the system completes the nascent relational schema by creating an empty resulting element or subgoal-slot. The slot can be filled-in with the help of menus (unit of measurement, label).


Inhalt von Vaters Kanne
15
1
Teil von Vaters Kanne
5
Anzahl der Füllungen
12
Wasser im Fass nach dem Giessen
24
1
Inhalt von Simons Kännchen
1
Inhalt von Simons Kännchen
Inhalt beider Kannen zusammen
Wasser im Fass vor dem Giessen
Wasser das der Vater vergiesst
Wasser das Simon vergiesst
alles Wasser das vergossen wird
verdunstetes Wasser
Der kleine Simon und sein Vater giessen den Gemüsegarten . Die Kanne des Vaters fasst 15 l . Simons Kännchen fasst den fünften Teil davon . Jeder füllt seine Kanne 12 mal . Danach sind inner noch 24 l Wasser im Regenfass . Wie viele l fasst das Regenfass ?
TEXT
Ende

Little Simon and his father water their vegetable garden. The father has a 15-liter can. Simon's can holds one fifth of that. Both fill their cans 12 times. After that there are still 24 liters in the waterbutt. How much water can the waterbutt hold?

Figure 4

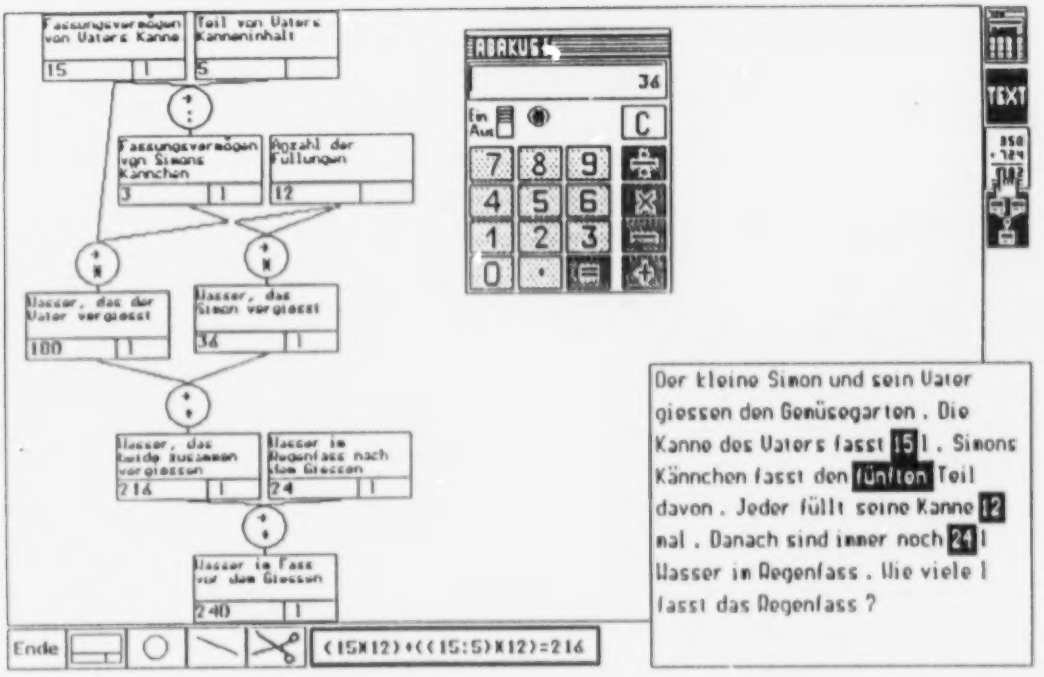
Partial solution tree



Little Simon and his father water their vegetable garden. The father has a 15-liter can. Simon's can holds one fifth of that. Both fill their cans 12 times. After that there are still 24 liters in the waterbutt. How much water can the waterbutt hold?

Figure 5

Complete solution tree


Fassungsvermögen von Vaters Kanne
15 l
Teil von Vaters Kanneninhalt
5
Fassungsvermögen von Simons Kännchen
3 l
Anzahl der Füllungen
12
Wasser, das der Vater vergiesst
180 l
Wasser, das Simon vergiesst
36 l
Wasser, das beide zusammen vergiessen
216 l
Wasser im Regenfass nach dem Giessen
24 l
Wasser im Fass vor dem Giessen
240 l
ABAKUS
36
TEXT
Der kleine Simon und sein Vater giessen den Gemüsegarten. Die Kanne des Vaters fasst 15 l. Simons Kännchen fasst den fünften Teil davon. Jeder füllt seine Kanne 12 mal. Danach sind immer noch 24 l Wasser in Regenfass. Wie viele l fasst das Regenfass ?
Ende
(15*12)+((15:5)*12)=216

Little Simon and his father water their vegetable garden. The father has a 15-liter can. Simon's can holds one fifth of that. Both fill their cans 12 times. After that there are still 24 liters in the waterbutt. How much water can the waterbutt hold?

Figure 6

Another variant of a complete solution tree

In order to *generate equations* from the solution tree, the student can click the right mouse-button on any numerical value in the tree. Then, the system displays the equation which generates the selected value. E.g. To click on the value "216" in Figure 5 will produce the partial equation

$$((15 : 5) + 15) * 12 = 216$$

while clicking on the same value in Figure 6 will produce the following equation:

$$((15 : 5) * 12) + (15 * 12) = 216.$$

And to click on the final value of "240" in Figure 5 will generate the answer-generating equation

$$((15 + (15:5)) * 12) + 24 = 240.$$

Figures 7 and 8 show, as a second illustrative example, a nascent version and the complete solution tree of a much more complex problem (Table 2).[17] The figures may give an impression of the flexibility of the system interface allowing the student to decompose a problem into relational building blocks or subgoals, and to construct solution trees of almost any complexity and in an arbitrary sequence in which subgoals are achieved.

Empirical observations

HERON is capable to record a log file of the order of activities performed on the screen (sequence, time characteristics, turn-taking of the mouse).[18] So far, 25 students from grade levels 3 to 6 have worked for one to two hours with the system. The students were working, with a few exceptions, in pairs of two. The main result from these tryouts is that after an instructional scaffolding phase of about ten to twenty minutes, including the construction of a sample solution tree, all pairs of students were able to construct on their own the solution tree of a given problem. Some further observations were:

---

[17] The problem was designed by Aebli et al. 1986 and thoroughly analysed by Staub 1987.

[18] The log file allows to do a *replay of the entire comprehension (solution) path* (Xerox-Version of HERON). This shall allow us, in a planned study, to get retrospective reports from the students on what they were thinking while comprehending and solving the problem.

Problem Text

(Afghanistan is a mountainous country, just like Switzerland, but much drier. There are fewer springs and streams, and water is rare and valuable. From a small village, a muletrack leads over a pass into a small town. To drive animals into town, one needs several days. At the end of the first day, one comes to an alp. Here a shepherd boy keeps 18 sheep and 15 goats. Above the alp, there is a snowfield which melts in the course of the summer. The water then flows towards the alp and gives 110 litres of drinking water a day for the animals. But now, the snow does not melt yet. Until then, the animals drink from a waterhole below the rocks, in which the water from a spring is collected. The hole is not leak-proof, so 35 litres of water ooze away every day. Each sheep drinks about 5,4 litres a day, each goat 3,8 litres. The shepherd drinks about 2,3 litres of goatmilk a day. The spring yields 350 litres of water a day. 5 litres evaporate every day. Each evening the shepherd drains the waterhole. He leads all the water that is present through a small canal into a basin. He just did that this evening. He estimates that there are 90 litres of water in the basin at the moment. Another 120 litres of water ooze away as it is being led to the basin. It is not lost, however. It irrigates the meadow below the small canal. As a result, more grass grows here. The shepherd can cut it 3 times a year and obtains 158 kilograms of hay each time. The hay is brought to the village and fed to the animals during wintertime. The basin is barred away with poles, so the sheep and the goats cannot drink from it. From time to time, the peasants drive some cows from the village over a pass into the town and sell them on the market. On the first day, they find water on the alp only. When they get there, the shepherd boy takes the poles from the basin and lets the cows drink from it. On the day on which he has estimated the available water, his father arrives at the alp late in the evening. He says to his son: <I want to bring 24 cows to the market as soon as possible. I think that they need a total of 320 litres of water up here.> The boy says: <I can calculate when you can come.> When can he come with the cows?)

Table 2: Text of a more complex situation problem.

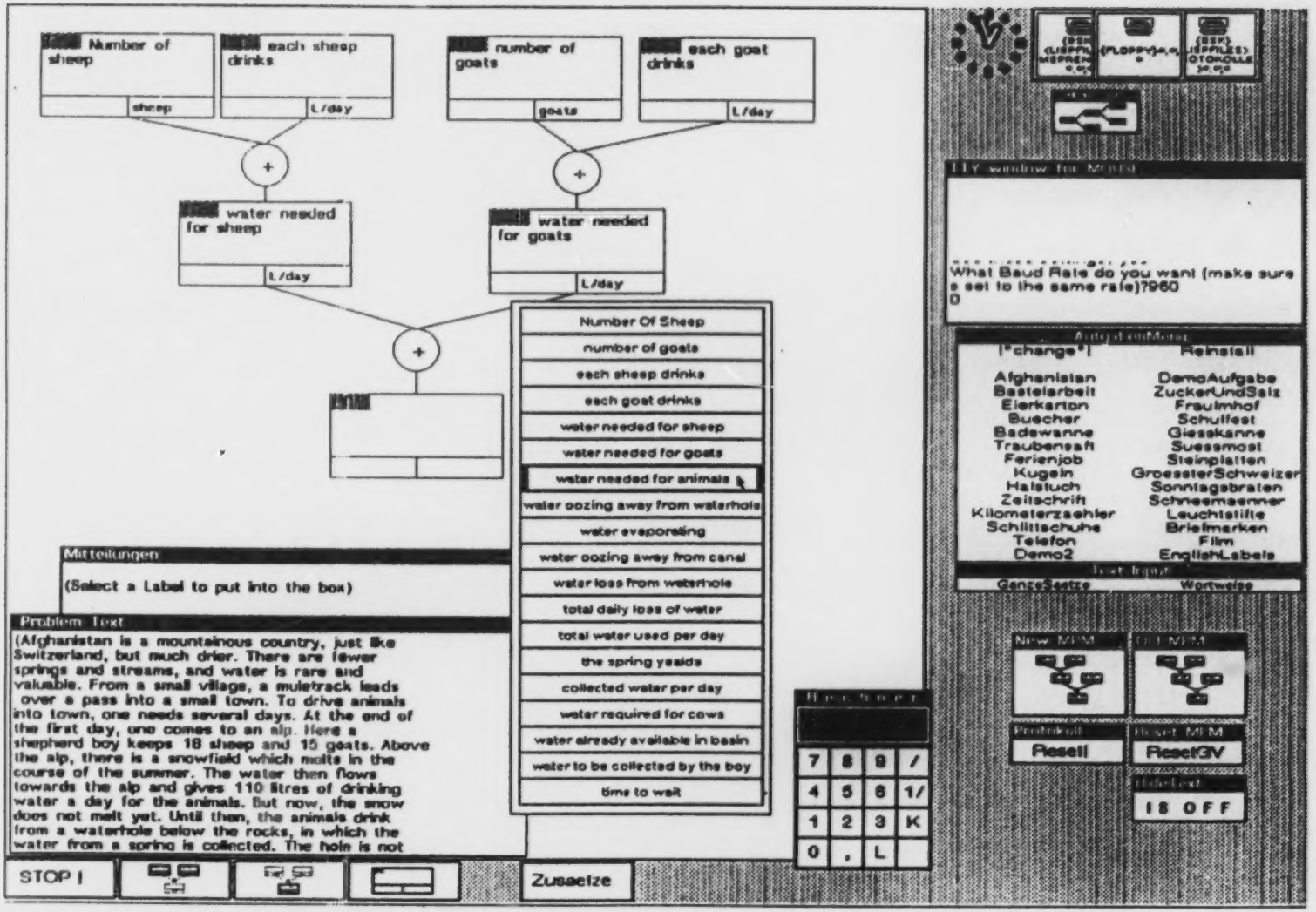

Number of sheep
sheep
each sheep drinks
L/day
number of goats
goats
each goat drinks
L/day
water needed for sheep
L/day
water needed for goats
L/day
Number Of Sheep
number of goats
each sheep drinks
each goat drinks
water needed for sheep
water needed for goats
water needed for animals
water oozing away from waterhole
water evaporating
water oozing away from canal
water loss from waterhole
total daily loss of water
total water used per day
the spring yields
collected water per day
water required for cows
water already available in basin
water to be collected by the boy
time to wait
Mitteilungen
(Select a Label to put into the box)
Problem Text
(Afghanistan is a mountainous country, just like Switzerland, but much drier. There are fewer springs and streams, and water is rare and valuable. From a small village, a muletrack leads over a pass into a small town. To drive animals into town, one needs several days. At the end of the first day, one comes to an alp. Here a shepherd boy keeps 18 sheep and 15 goats. Above the alp, there is a snowfield which melts in the course of the summer. The water then flows towards the alp and gives 110 litres of drinking water a day for the animals. But now, the snow does not melt yet. Until then, the animals drink from a waterhole below the rocks, in which the water from a spring is collected. The hole is not
STOP !
Zusaetze
What Baud Rate do you want (make sure
What Baud Rate do you want (make sure e set to the same rate)?960
0
Afghanistan
Bastelarbeit
Eierkarton
Buecher
Badewanne
Traubensaft
Ferienjob
Kugeln
Halstuch
Zeitschrift
Kilometerzaehler
Schlittschuhe
Telefon
Demo2
DemoAufgabe
ZuckerUndSalz
Frauimhof
Schulfest
Giesskanne
Suessmost
Steinplatten
GroesserSchweizer
Sonntagsbraten
Schneemaenner
Leuchtstifte
Briefmarken
Film
EnglishLabels
GanzeSaetze
Wortweise
New MFM
Off MFM
Reset MFM
Reset3V
IS OFF

Figure 7. Nascent solution tree of the Afghanistan task.

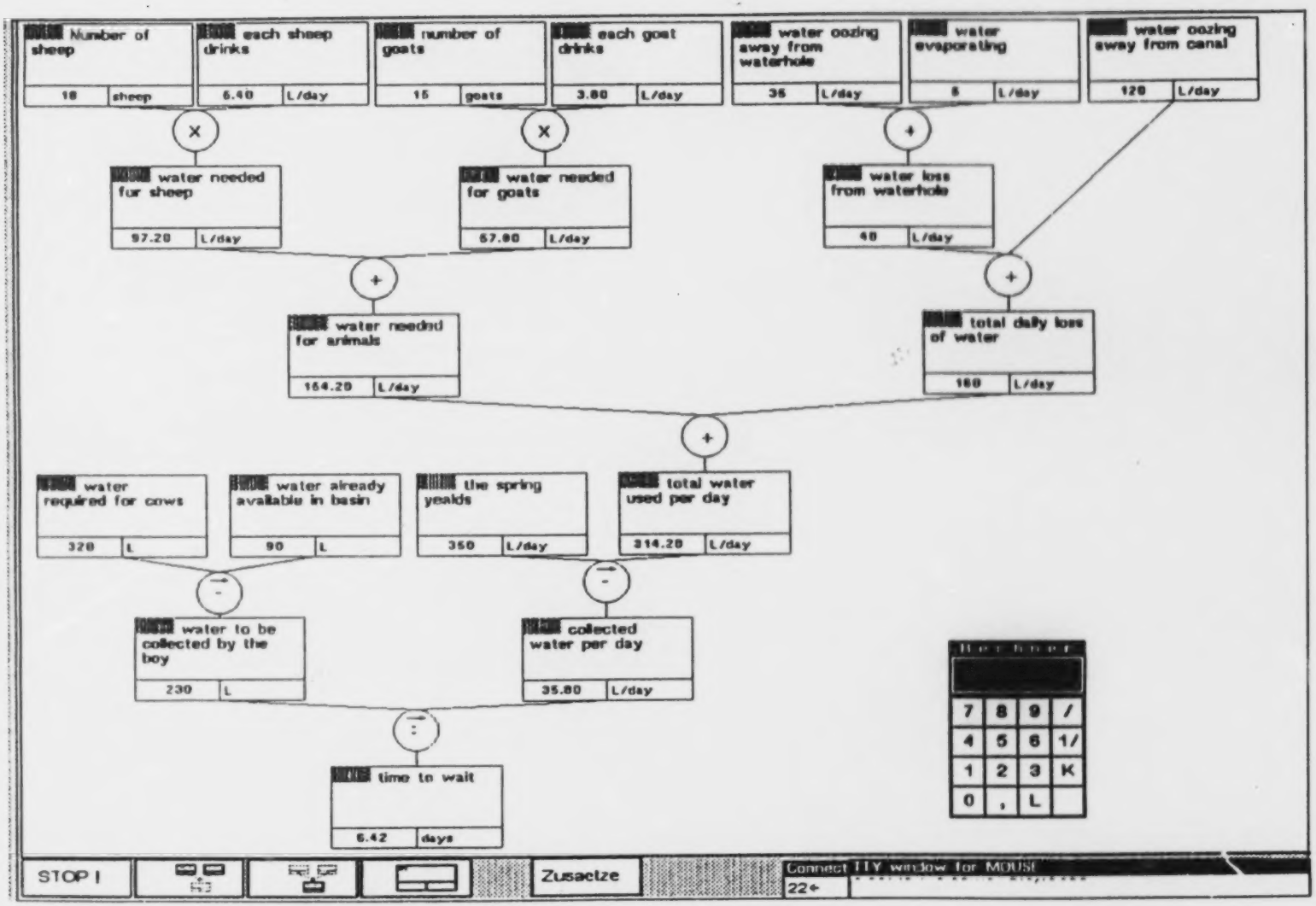


Figure 8. Complete solution tree of the Afghanistan task.

- On the average, students switched 10 times between the problem text and the tree being constructed (range from 1 to 32);

- Activation of the problem text occured almost regularily after the completion of a relational schema (going back to the text after a subgoal has been achieved);

- Most students followed a *modus operandi* similar to the one described by Derry & Hawkes (1989): after a first relational unit is schematically constructed (by drawing three boxes or situation units), the boxes are numerically and situationally interpreted by filling in the numbers (one of them to be computed), the units of measurement and the labels; then, the next schema is constructed including the previous resulting set as one of its elements. The construction goes on the same way until the tree is completed;

- All students (but one) followed a "reflexive forward-chaining" strategy (Derry & Hawkes 1989, 11) which stepwise proceeds from the given information to the anticipated goal state;

- While solving one task, there were, on the average, 12 turn-takings of the mouse between the pair of students;

- The largest error catagories observed were operation errors and labeling errors.[19]

Feedback Capabilities

HERON's basic representation of mathematical word problems is a propositional micro-structure similar to a textbase sensu Kintsch (1974), or to the initial problem representation used in the simulation model of Reusser (1987). Since it is possible to store a very dense network of problem information in this propositional structure,

[19] These error categories were also high in the error study of Derry & Hawkes (1989)

it should be feasible, in principle, to monitor the student's analysis of the problem text, as well as his construction of a sensible mathematical problem model.

However, so far HERON's intelligence is modest. It does not know anything about student models, i.e. about the thinking of individual students while solving a problem.[20] But HERON "knows" how to build conceptual networks (tree structures), and what students are *doing* when constructing solution trees. Two ways of including capabilities of providing feedback on operation and on labeling errors (text labels, units of measurement) are currently under exploration. The first approach is functionally similar to the approach of Derry & Hawkes (1989) treating errors as deviations from recognizable solution paths which are predefined for the system. The second approach uses a component that we call graph-completion. Based on the mathematical law of distribution it is possible, given one single solution tree to the system, to infer the complete solution space from which, again, errors can be detected as deviations.

Based on preliminary explorations, both ways of error-handlung seem to be promising. HERON will shortly be able to provide specific feedback if

- two situation units are connected by a false operation,
- a labeling error occurs,
- a false measurement unit is selected,
- if a piece of relevant numerical information is omitted and not included in the solution tree,
- if a piece of irrelevant information is included in the solution tree.

---

[20] This does not mean that we do not intend to customize our system to the actual needs of students. But it means that information about the ability level and about the learning of students is not collected by methods of *simulating their cognitive behavior*. Based on the recorded log file data about the overt construction processes of students, including error patterns, success and failure rates related to specific types of problems, we are optimistic about the feasibility of achieving some adaptive power of the system.

Ongoing work

HERON records in log file all activities performed on the screen, including time characteristics. This allows to do replays of comprehension and solution paths. Pairs of students can think about and discuss their solution paths. The experimenter can use them for doing a *self-confrontation* in order to elicit more complete retrospective reports from the students.

In order to successfully construct solution trees, students must understand the domain-specific relations which are implied by particular problem typed, e.g. the relations between time, distance and rate, or between capital investment, interest rate and annual profit. Because students often have difficulties to flexibly operate within these schemata, HERON provides a training module for relational schemata, where the student can exercise his understanding of these relations in a simple interface setting. E.g. the student is asked to combine or to complete triples of domain concepts and to put them into correct mathematical relations (after having selected an unknown; Figures 9 a,b). This training module - there are others under development - underlines the design of the system as to become a cognitive toolbox assisting students in a diversity of ways in understanding and solving a wide class of problems.

In an ongoing study, HERON will be used in a classroom setting, as a part of a didactical learning environment, which is intended to support collaborative problem solving. From collaborative problem solving, especially in groups or pairs, we expect two things: (1) positive effects on the metacognitive and strategic quality of the comprehension processes involved in solving word problems, and (2) an improvement of the efficiancy of the use of the system. Pairs of students working together can help and support each other in many ways: not only in the activities directly related to the solving of the problems at hand, but also in the more effective use of the computer system. Pairs of students who monitor each other's behavior not only can provide mutual feedback and hints but also may contribute to the improvment of the overall effectivity - especially with regard to error handling - of the system. Computer-based learning environments are still very weak and require multiple and combined expertise - from the system's own (modest) intelligence and from intelligent users.

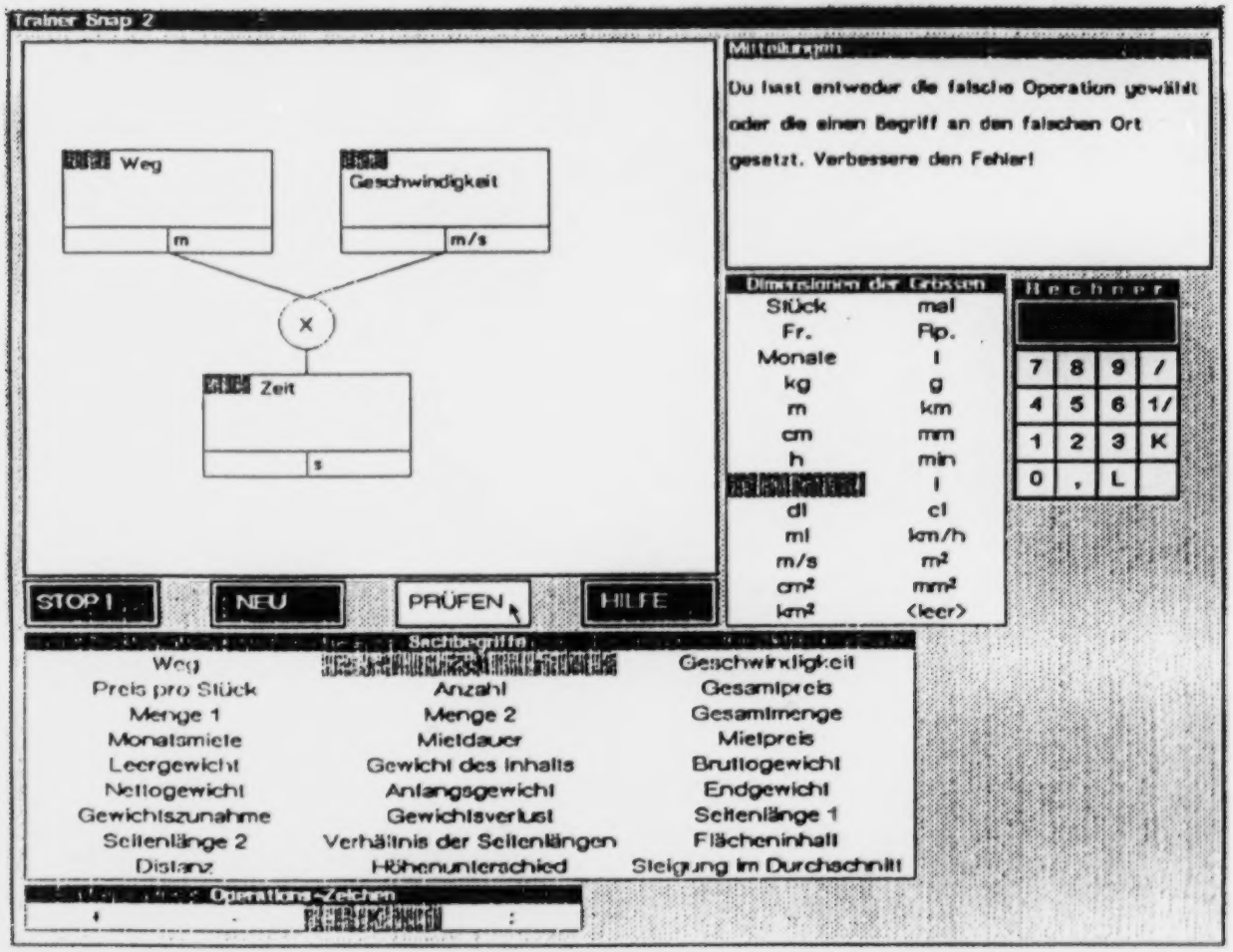


Figure 9a

Training module for relational schemata.
The example shows a wrongly selected operation with the corresponding feedback of the system

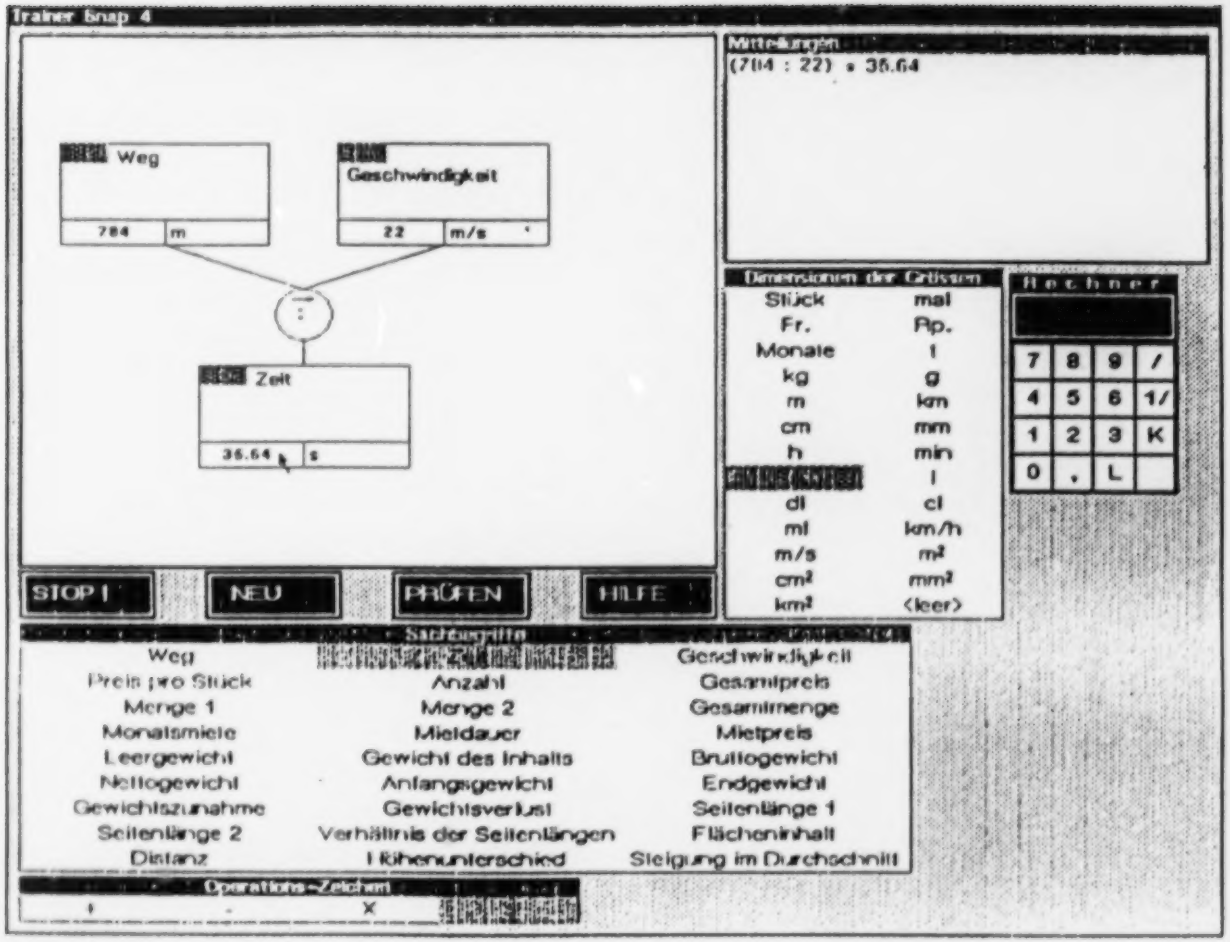


Figure 9b

Training module for relational schemata.
The example shows a correct schema including numeric values

References

AEBLI, H. (1961) *Grundformen des Lehrens.* Stuttgart: Klett

AEBLI, H. (1980) *Denken. Das Ordnen des Tuns. Band 1. Stuttgart: Klett*

AEBLI, H., RUTHEMANN, U. & STAUB, F. (1986) Sind Regeln des Problemlösens lehrbar? *Zeitschrift für Pädagogik. 32*, 617-638)

ANDERSON, J.A. (1989) *Psychology and Intelligent tutoring.* Paper presented at the fourth International Conference on AI in Education. Amsterdam.

ANDERSON, J.A., BOYLE, C.F., FARRELL, R. REISER, B. (1984) *Cognitive principles in the design of computer tutors. Proceddings of the Sixth Annual Conference of the Cognitive Science Society*, Boulder, Colorado.

BECK, E. (1989) *Eigenständiges Lernen. Beiträge zur Lehrerbildung. 7, 169-178.*

BOBROW, ,D.G.(1964) *Natural language input for a computer problem solving system.*- Doctoral thesis. MIT

BREIDENBACH, W. (1963) *Rechnen in der Volksschule.* Hannover: Schroedel Verlag.

BROWN, A.L. & PALINCSAR, A.S. (1989) Guided, cooperative learning and individual knowledge acquisition. In L.B. RESNICK (ED.) *Knowing, learnning and instruction.* Hillsdale, NJ: Erlbaum.

BRUNER, J. (1961) The act of discovery. *Harvard Educational Review. 31*, 21-32.

BRUNER, J.S. (1966) *Studien zur kognitiven Entwicklung. Stuttgart:Klett.*

BRUNER, J. (1986) *Actual minds, possible worlds.* Cambridge, MA: Harvard University Press.

CLANCEY, W. (1988) The knowledge engineer as student: metacognitive basis for asking good questions. In H. MANDL & A. LESGOLD (Eds.) *Learning issues for intelligent tutoring systems.* New York: Springer.

COLLINS, A. & BROWN, J.S. (1988) The computer as a tool for learning through reflection. In H. MANDL & A. LESGOLD (Eds.) *Learning issues for intelligent tutoring systems.* New York: Springer.

COLLINS, A., BROWN, J.S. & NEWMAN, S.E. (1989) *Cognitive apprenticeship: Teaching the crafts of reading, writing, and mathematics.* In L.B. RESNICK (Ed.) *Knowing, learning and instruction.* Hillsdale, NJ: Erlbaum.

COMENIUS, AMOS (1653) *Orbis pictus.*

CUMMINS, D., KINTSCH, W., REUSSER, K. & WEIMER, R. (1988) *The role of understanding in solving word problems. Cognitive Psychology. 20, 439-462.*

DE CORTE, E., VERSCHAFFEL, L. & DE WIN, L. (1985) Influence of rewording problems on children's problem represenations and solutions. *Journal of Educational Psychology, 77*, 460-470.

DERRY, S. (1989) Strategy and expertise in solving word problems. In CH.B. MCCORMICK, G. MILLER, & M. PRESSLEY (Eds.) *Cognitive strategy research: From basic research to educational applications.* New York: Springer.

DERRY, S.J. & HAWKES,L.W. (1989) *Error-driven cognitive apprenticeship: a feasible ITS approach.* Florida State University, Tallahassee.

FRICKE, A. Sachrechnen. *Das Lösen angewandter Aufgaben.* Stuttgart: Klett.

HUDSON, T. (1983) Correspondences and numerical differences between disjoint sets. *Child Development, 54*, 84-80.

HUTCHINS, E.L., HOLLAN, J.D. & NORMAN, D.A.(1985) Direct manipulation interfaces. In D.A. NORMAN and S.W. DRAPER (Eds.) *User centered system design: New perspectives on human-computer interaction.* Hilssdale, N.J.: Erlbaum.

KINTSCH, W. (1974) *The representation of meaning in memory.* Hillsdale, N.J.: Erlbaum.

KINTSCH, W. (1989) A theory of discourse comprehension: Implicatioons for a tutor for word algebra problems. Paper presented at the Third European Conference for Research on Learning and Instruction. Madrid, September 3-8.

KINTSCH, W. & GREENO, J.G. (1985) Understanding and solving word arithmetic problems. *Psychological Review, 92*, 109-129.

LARKIN, J.M., MCDERMOTT, J., SIMON, D.P., & SIMON, H.A. (1980) Models of competence in solving physics word problems. *Cognitive Science, 4*, 317-345.

MEAD, G.H. (1934) *Mind, self, and society*. Chicago.

NATHAN, J.N., KINTSCH, W. & LEWIS, C. (1988) *Tutoring Algebra word problems.* Institute of Cognitive Science, Uni-versity of Colorado, Boulder, Technical Report 88-12.

NATHAN, J.N., KINTSCH, W., YOUNG, E. (1990) *A theory of algebra word problem comprehension and its implications for unintelligent tutoring systems.* Institute of Cognitive Science, University of Colorado, Boulder, Technical Report 90-02.

OHLSSON, S. (1986) Some principles of intelligent tutoring. *Instructional Science, 14*, 293-326.

PALINCSAR, A.S. & BROWN, A.L. (1984) Reciprocal teaching of comprehension-fostering and comprehension-monitoring activities. *Cognition and Instruction, 1*, 117-175.

PIAGET, J. (1950) *Die Entwicklung des Erkennens, Band 1*. Stuttgart: Klett.

PIAGET, J. (1973) *To understand is to invent.* New York: Grossman.

RESNICK, L.B. & JOHNSON, A. (1989) *Intelligent machines for intelligent people: Cognitive theory and the future of computer-assisted learning*. University of Pittsburgh.

REUSSER, K. (1984) *Problemlösen in wissenstheoretischer Sicht*. Dissertation. Univerität Bern.

REUSSER, K. (1985) *From text to situation to equation*. Technical Report. Institute of Cognitive Science. University of Colorado, Boulder.

REUSSER, K. (1987) *The computer's effort after meaning: Word problem solving as a process of text comprehension and mathematization*. Forschungsbericht Nr. 4. Universität Bern: Abteilung Pädagogische Psychologie.

REUSSER, K. (1989a) *Textual and situational factors in mathematical word problems*. Paper presented at the Third Conference of the European Association for Research on Learning and Instruction. September. Madrid/Spain.

REUSSER, K. (1989b) *Vom Text zur Situation zur Gleichung. Kognitive Simulation von Sprachverständnis und Mathematisierung beim Lösen von Textaufgaben*. Habilitationsschrift. Universität Bern.

REUSSER, K. (1990) From text to situation to equation: cognitive simulation of understanding and solving mathematical word problems. In: H. MANDL, E. DE CORTE, N. BENNETT, & H.F. FRIEDRICH, (Eds.)(1989) *Learning and Instruction in an International Context*. *Vol. II*. New York: Pergamon Press.

SCARDAMALIA, M. & BEREITER (1987) *Computer-supported intentional learning environments*. Toronto: Ontario Institute for the Study of Education.

SELZ, O. (1922) *Zur Psychologie des produktiven Denkens und des Irrtums*. Bonn: Cohen.

SLEEMAN, D. & BROWN, J.S. (1982) *Intelligent tutoring systems*. New York: Academic Press.

STAUB, F. (1987) *Diagnose des Verständnisses von semantisch komplexen mathematischen Textaufgaben anhand von Problemwiedergaben*. Lizentiatsarbeit. Universität Bern.

VAN DIJK, T. & KINTSCH, W. (1983) *Strategies of discourse comprehension*. New York: Academic Press.

VYGOTSKY, L.S. (1978) *Minds in society: The development of higher psychological process*. Cambridge, MA: Harvard University Press.

WINOGRAD, T. & FLORES, F. (1986) *Understanding computers and cognition. A new foundation for design*. Norwood, NJ: Ablex.